

Distributional Semantics

João Sedoc

IntroHLT class

November 4, 2019

Intuition of distributional word similarity

Nida example:

A bottle of ***tesgüino*** is on the table
Everybody likes ***tesgüino***
Tesgüino makes you drunk
We make ***tesgüino*** out of corn.

From context words humans can guess ***tesgüino*** means

- an alcoholic beverage like **beer**

Intuition for algorithm:

- Two words are similar if they have similar word contexts.

Distributional Hypothesis

If we consider **optometrist** and **eye-doctor** we find that, as our corpus of utterances grows, these two occur in almost the same environments. In contrast, there are many sentence environments in which **optometrist** occurs but **lawyer** does not...

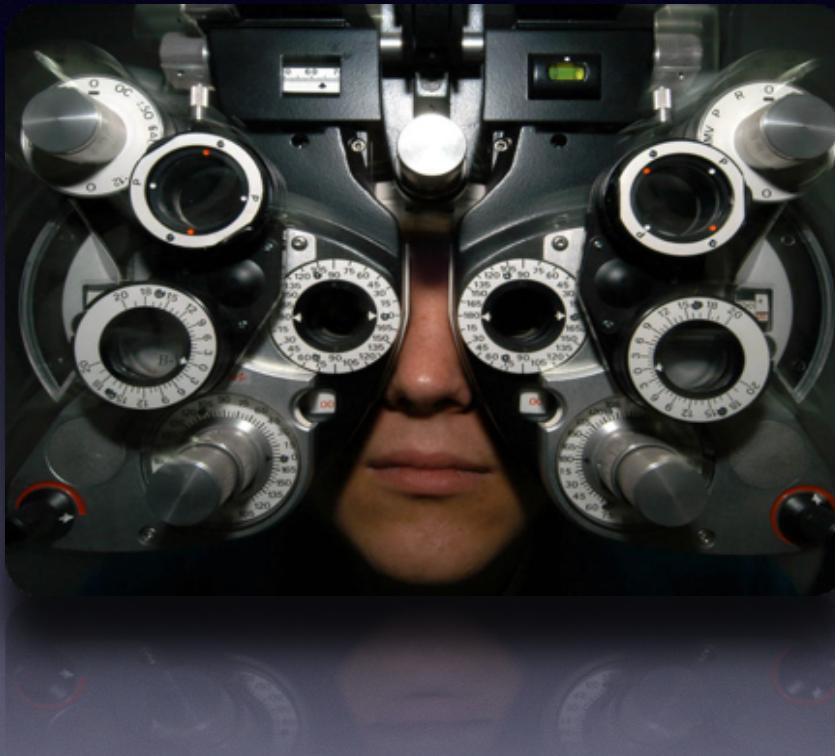
It is a question of the relative frequency of such environments, and of what we will obtain if we ask an informant to substitute any word he wishes for **optometrist** (not asking what words have the same meaning).

These and similar tests all measure the probability of particular environments occurring with particular elements... If A and B have almost identical environments we say that they are synonyms.

—Zellig Harris (1954)

“You shall know a word by the company it keeps!”

—John Firth (1957)



Distributional models of meaning
= vector-space models of meaning
= vector semantics

Intuitions: Zellig Harris (1954):

- “oculist and eye-doctor ... occur in almost the same environments”
- “If A and B have almost identical environments we say that they are synonyms.”

Firth (1957):

- “You shall know a word by the company it keeps!”

Intuition

Model the meaning of a word by “embedding” in a vector space.

The meaning of a word is a vector of numbers

- Vector models are also called “**embeddings**”.

Contrast: word meaning is represented in many computational linguistic applications by a vocabulary index (“word number 545”)

$$\text{vec}(\text{"dog"}) = (0.2, -0.3, 1.5, \dots)$$

$$\text{vec}(\text{"bites"}) = (0.5, 1.0, -0.4, \dots)$$

$$\text{vec}(\text{"man"}) = (-0.1, 2.3, -1.5, \dots)$$

Term-document matrix

Term-document matrix

Each cell: count of term t in a document d : $\text{tf}_{t,d}$:

- Each document is a count vector in \mathbb{N}^v : a column below

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Term-document matrix

Two documents are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

The words in a term-document matrix

Each word is a **count vector** in \mathbb{N}^D : a row below

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

The words in a term-document matrix

Two **words** are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Brilliant insight: Use running text as implicitly supervised training data!

- A word s near *fine*
 - Acts as gold ‘correct answer’ to the question
 - “Is word w likely to show up near *fine*? ”
- No need for hand-labeled supervision
- The idea comes from **neural language modeling**
 - Bengio et al. (2003)
 - Collobert et al. (2011)

Word2vec

Popular embedding method

Very fast to train

Code available on the web

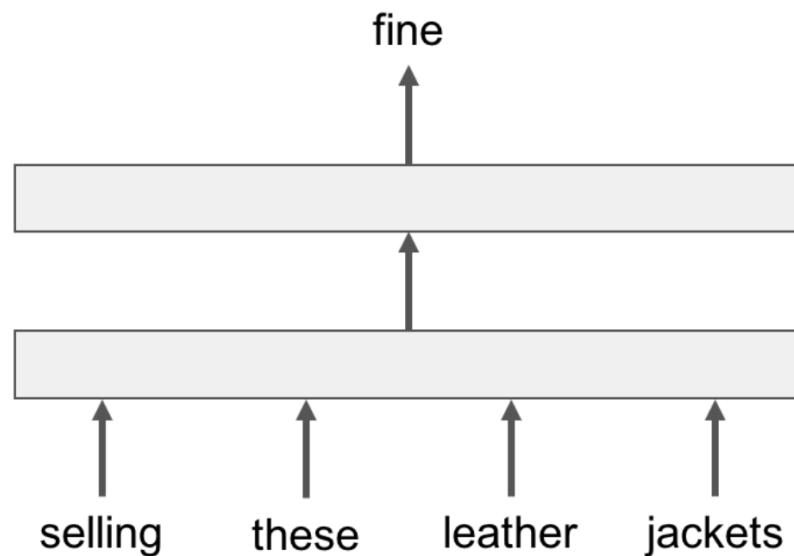
Idea: **predict** rather than **count**

Word2vec

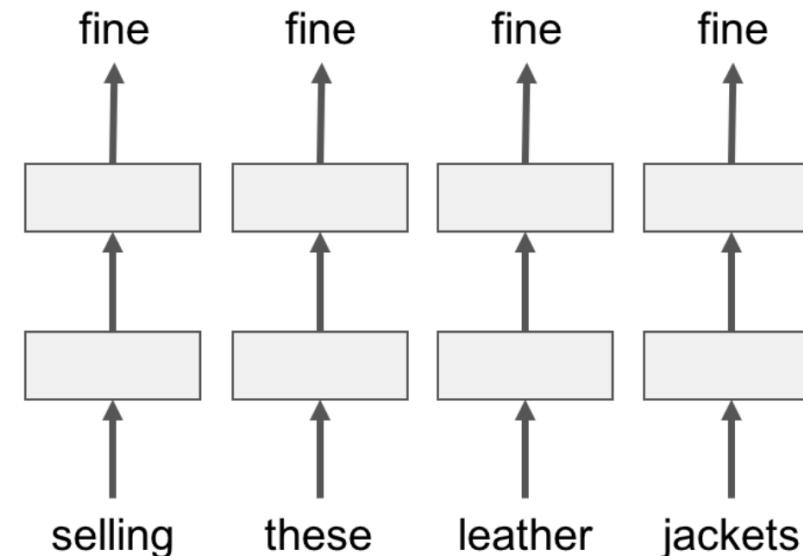
- Instead of **counting** how often each word w occurs near “*fine*”
- Train a classifier on a binary **prediction task**:
 - Is w likely to show up near “*fine*”?
- We don’t actually care about this task
 - But we’ll take the learned classifier weights as the word embeddings

Word2vec

CBOW



SKIPGRAM

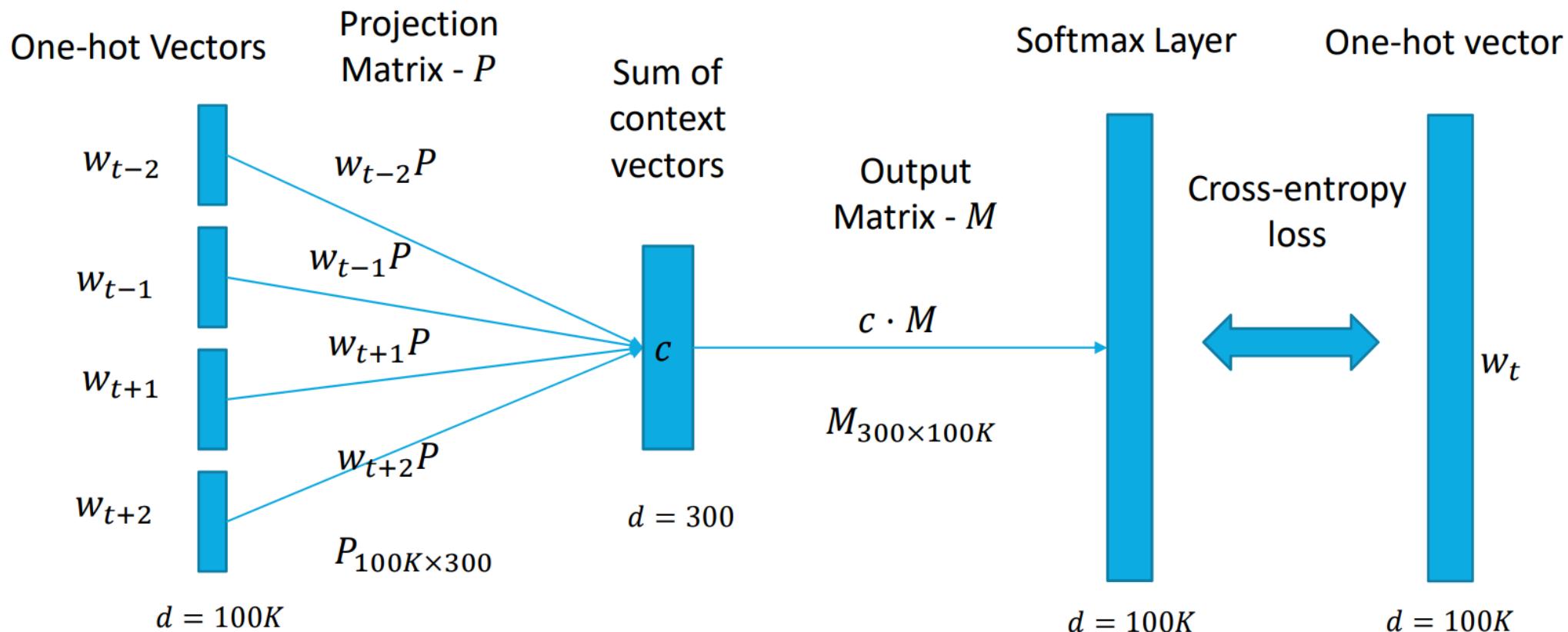


I am selling these fine leather jackets

CBOW – high level

The resulting projection matrix P is the embedding matrix

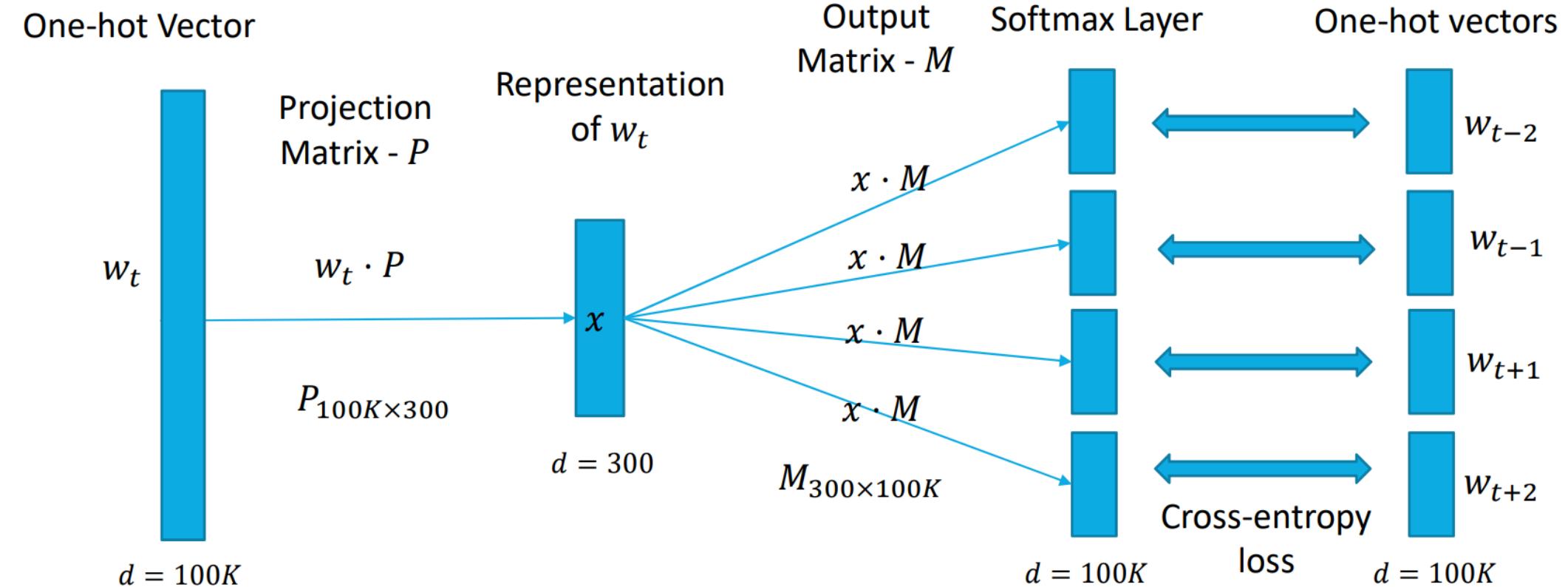
Goal: Predict the middle word given the words of the context



Skip-gram – high level

The resulting projection matrix P is the embedding matrix

Goal: Predict the context words given the middle word



Dense embeddings you can
download!



Word2vec (Mikolov et al.)

<https://code.google.com/archive/p/word2vec/>



Fasttext <http://www.fasttext.cc/>

Glove (Pennington, Socher, Manning)

<http://nlp.stanford.edu/projects/glove/>



Why vector models of meaning? Computing the similarity between words

“**fast**” is similar to “**rapid**”

“**tall**” is similar to “**height**”

Question answering:

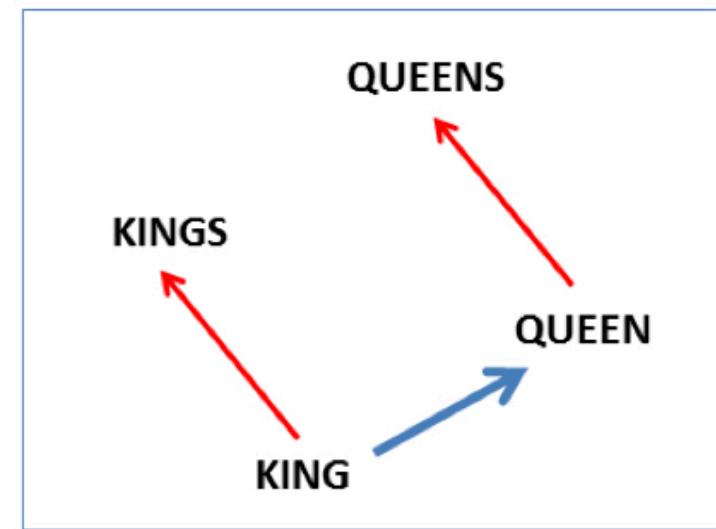
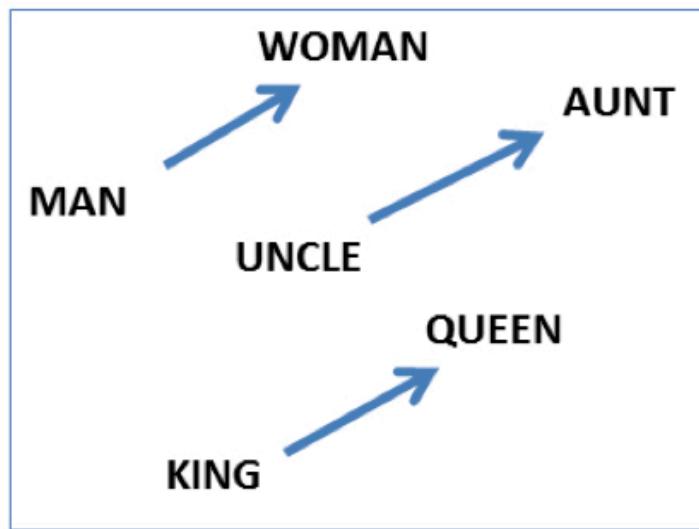
*Q: “How **tall** is Mt. Everest?”*

*Candidate A: “The official **height** of Mount Everest is 29029 feet”*

Analogy: Embeddings capture relational meaning!

$\text{vector('king')} - \text{vector('man')} + \text{vector('woman')} \approx \text{vector('queen')}$

$\text{vector('Paris')} - \text{vector('France')} + \text{vector('Italy')} \approx \text{vector('Rome')}$



Evaluating similarity

Extrinsic (task-based, end-to-end) Evaluation:

- Question Answering
- Spell Checking
- Essay grading

Intrinsic Evaluation:

- Correlation between algorithm and human word similarity ratings
 - Wordsim353: 353 noun pairs rated 0-10. $sim(plane,car)=5.77$
- Taking TOEFL multiple-choice vocabulary tests
- Levied is closest in meaning to:
imposed, believed, requested, correlated

Evaluating embeddings

Compare to human scores on word similarity-type tasks:

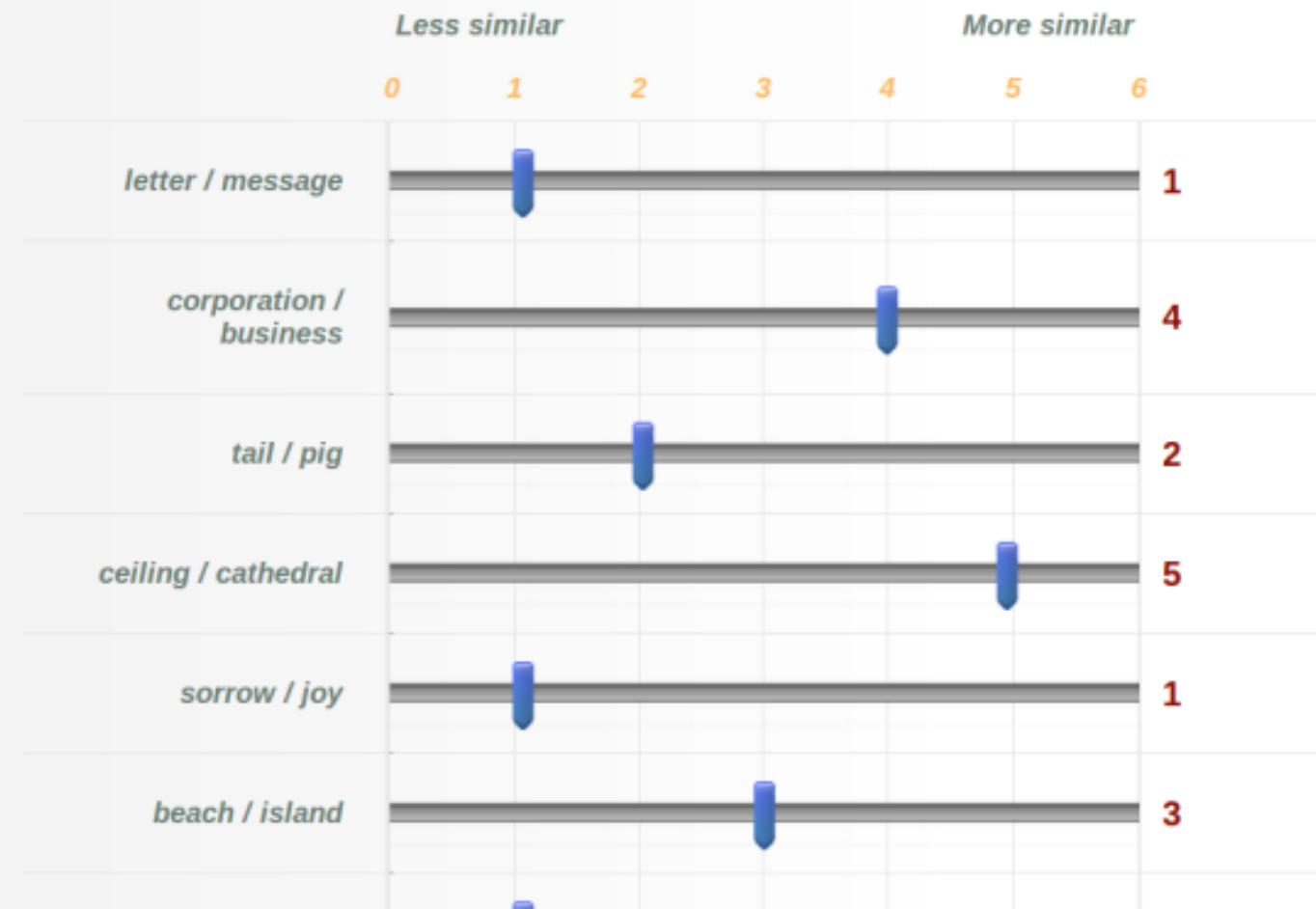
- WordSim-353 (Finkelstein et al., 2002)
- SimLex-999 (Hill et al., 2015)
- Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)
- TOEFL dataset: *Levied is closest in meaning to: imposed, believed, requested, correlated*

Intrinsic evaluation

cos sim	Psycho linguistic experiment		mean 10 judges	WordSim 353
	<u>↑</u>	<u>↓</u>		
	Love , sex		6.8	
:	tiger , cat		1.3	
:	tiger , tiger		10	
:	fertility , egg		6.7	
:	stock , egg		1.8	
	professor , cucumber		0.3	GOLD STANDARD

Intrinsic evaluation

Rate the following word pairs according to how similar they are by moving the slider



Measuring similarity

Given 2 target words v and w

We'll need a way to measure their similarity.

Most measure of vectors similarity are based on the:

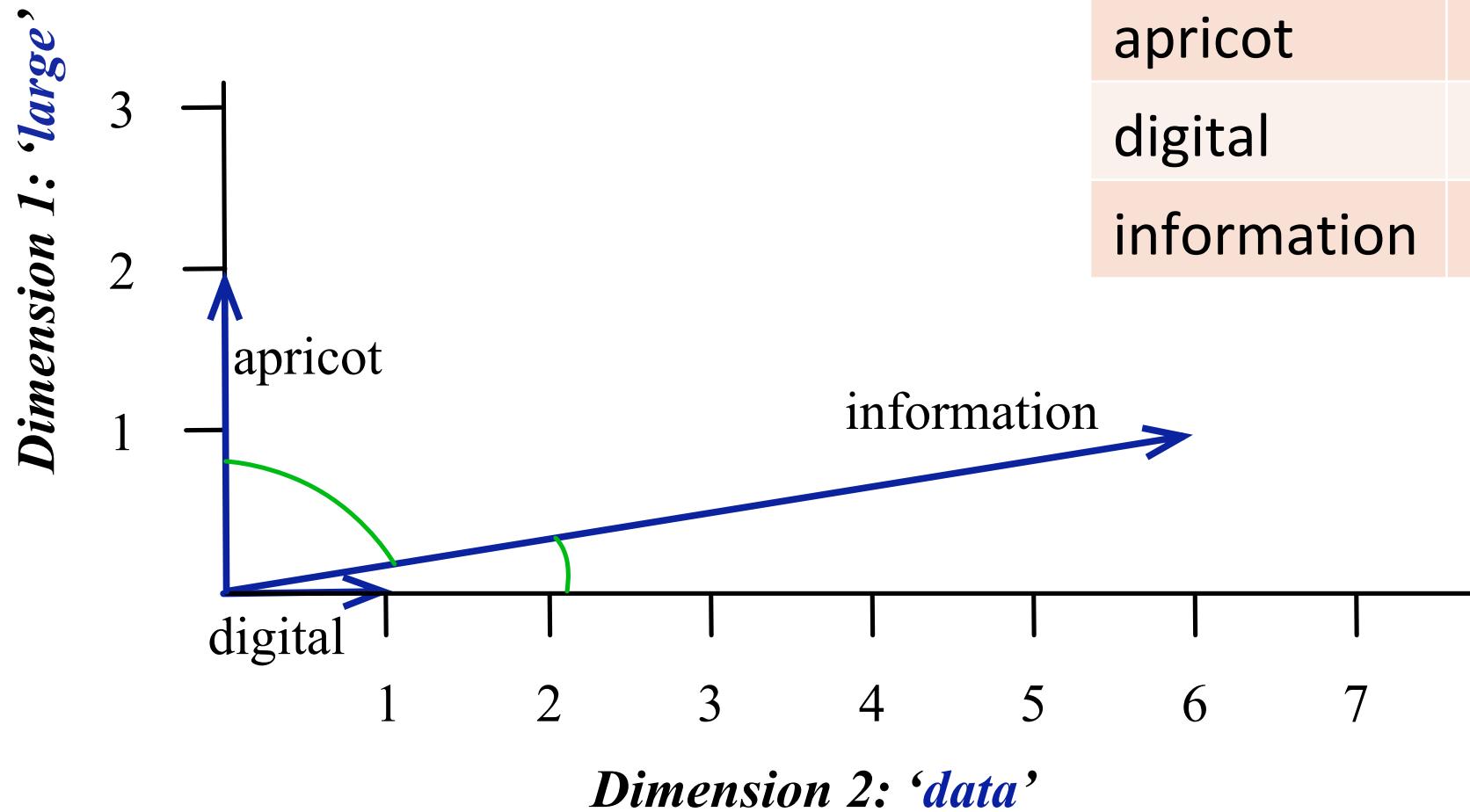
Cosine between embeddings!

The similarity between two vectors v and w is:

$$\frac{v \cdot w}{\|v\| \|w\|}$$

- High when two vectors have large values in same dimensions.
- Low (in fact 0) for **orthogonal vectors** with zeros in complementary distribution

Visualizing vectors and angles



	large	data
apricot	2	0
digital	0	1
information	1	6

Bias in Word Embeddings

Extreme *she* occupations

- | | | |
|-----------------|-----------------------|------------------------|
| 1. homemaker | 2. nurse | 3. receptionist |
| 4. librarian | 5. socialite | 6. hairdresser |
| 7. nanny | 8. bookkeeper | 9. stylist |
| 10. housekeeper | 11. interior designer | 12. guidance counselor |

Extreme *he* occupations

- | | | |
|----------------|------------------|----------------|
| 1. maestro | 2. skipper | 3. protege |
| 4. philosopher | 5. captain | 6. architect |
| 7. financier | 8. warrior | 9. broadcaster |
| 10. magician | 11. figher pilot | 12. boss |

More to come on bias later ...

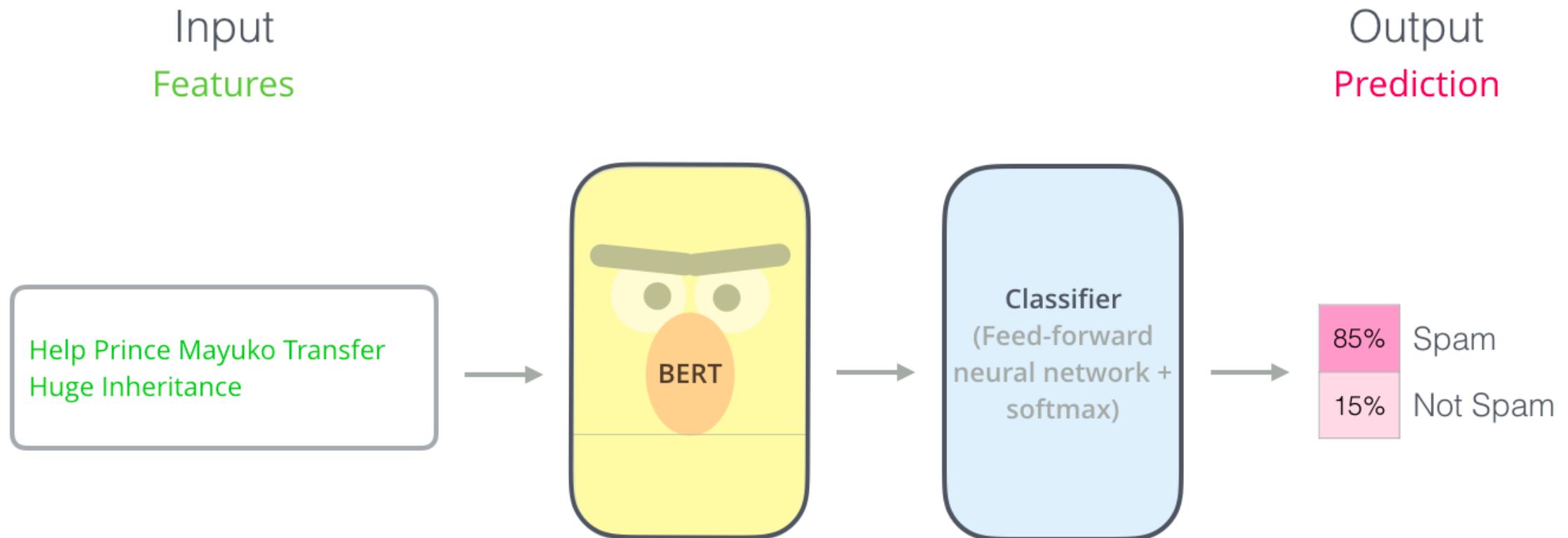
Embeddings are the workhorse of NLP

- Used as pre-initialization for language models, neural MT, classification, NER systems...
- Downloaded and easily trainable
- Available in ~100s of languages
- And ... **contextualized word embeddings** are built on top of them

Contextualized Word Embeddings

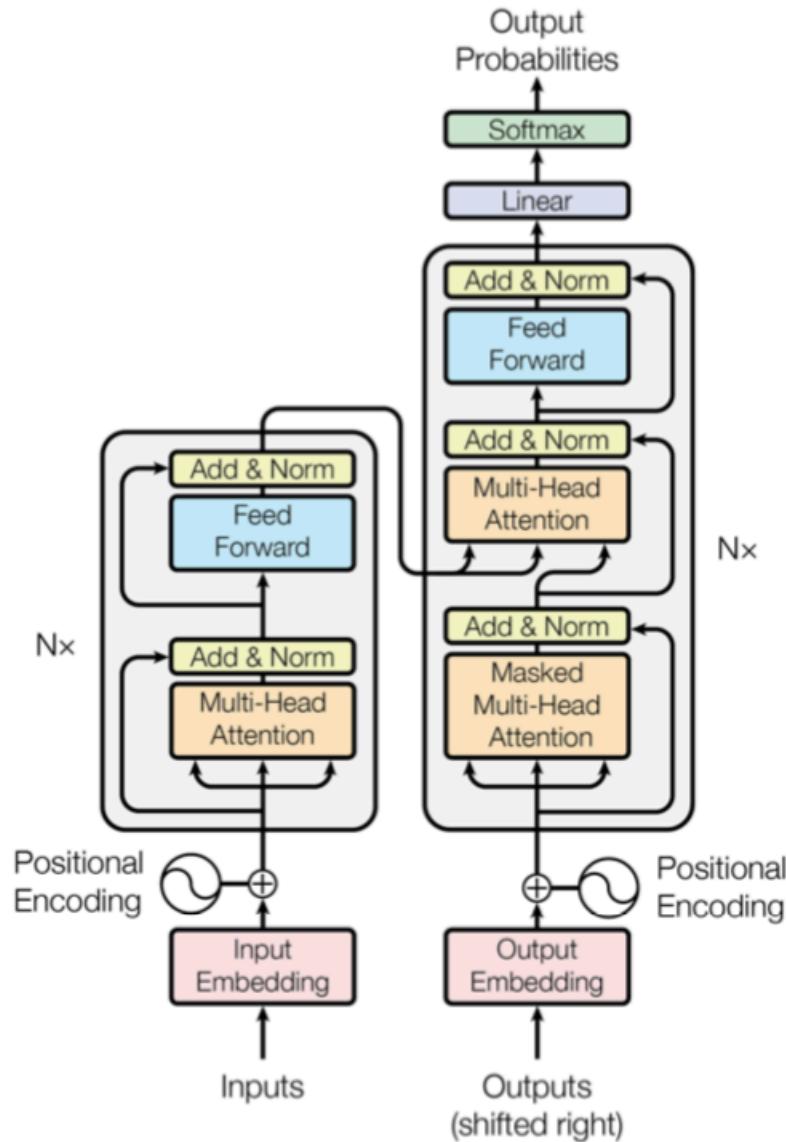


BERT - Deep Bidirectional Transformers

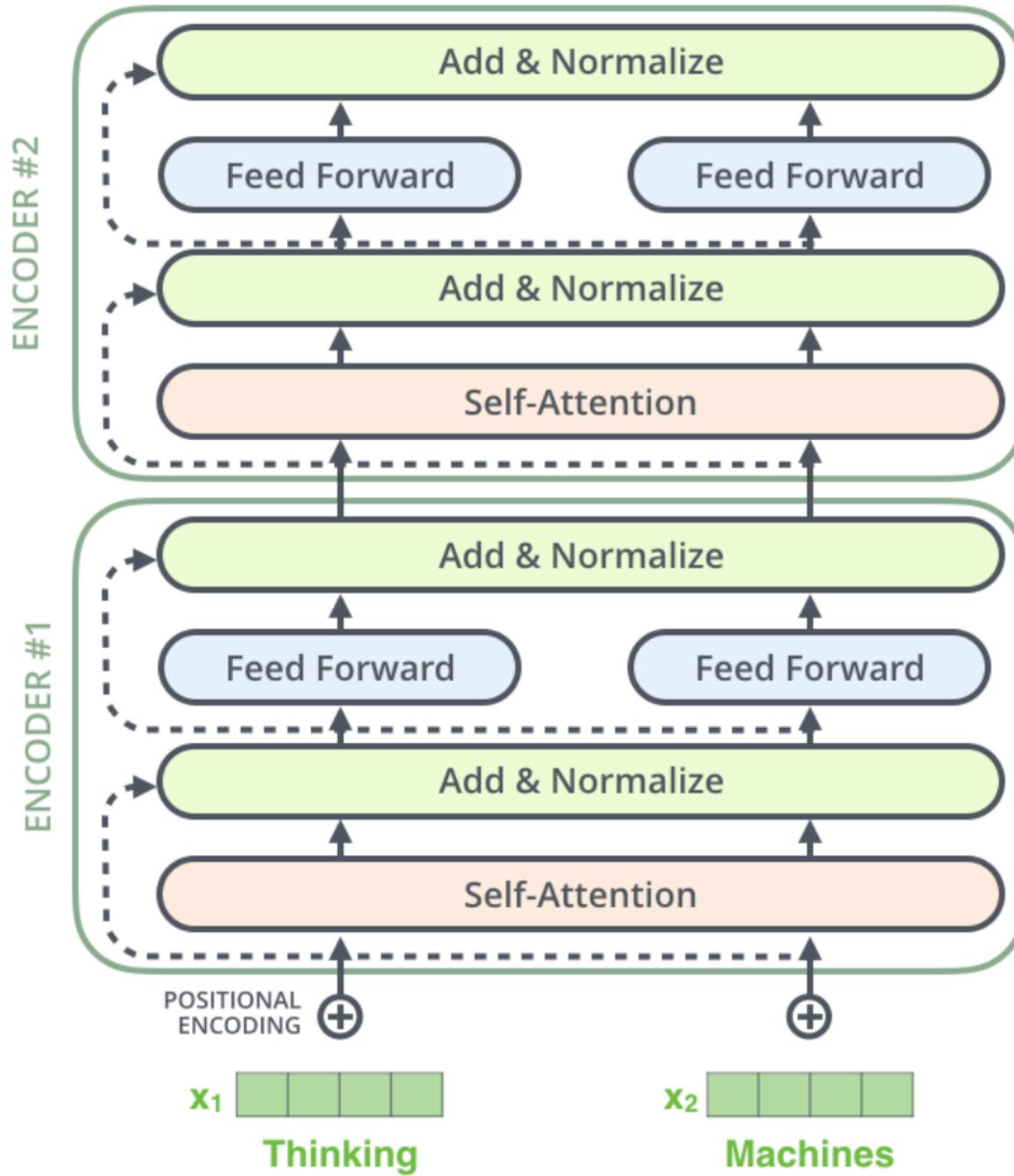


From last time... Putting it together

- Multiple (N) layers
- For encoder-decoder attention, Q: previous decoder layer, K and V: output of encoder
- For encoder self-attention, Q/K/V all come from previous encoder layer
- For decoder self-attention, allow each position to attend to all positions up to that position
- Positional encoding for word order



Transformer



Training BERT

- BERT has two training objectives:
 1. Predict missing (masked) words
 2. Predict if a sentence is the next sentence

BERT- predict missing words

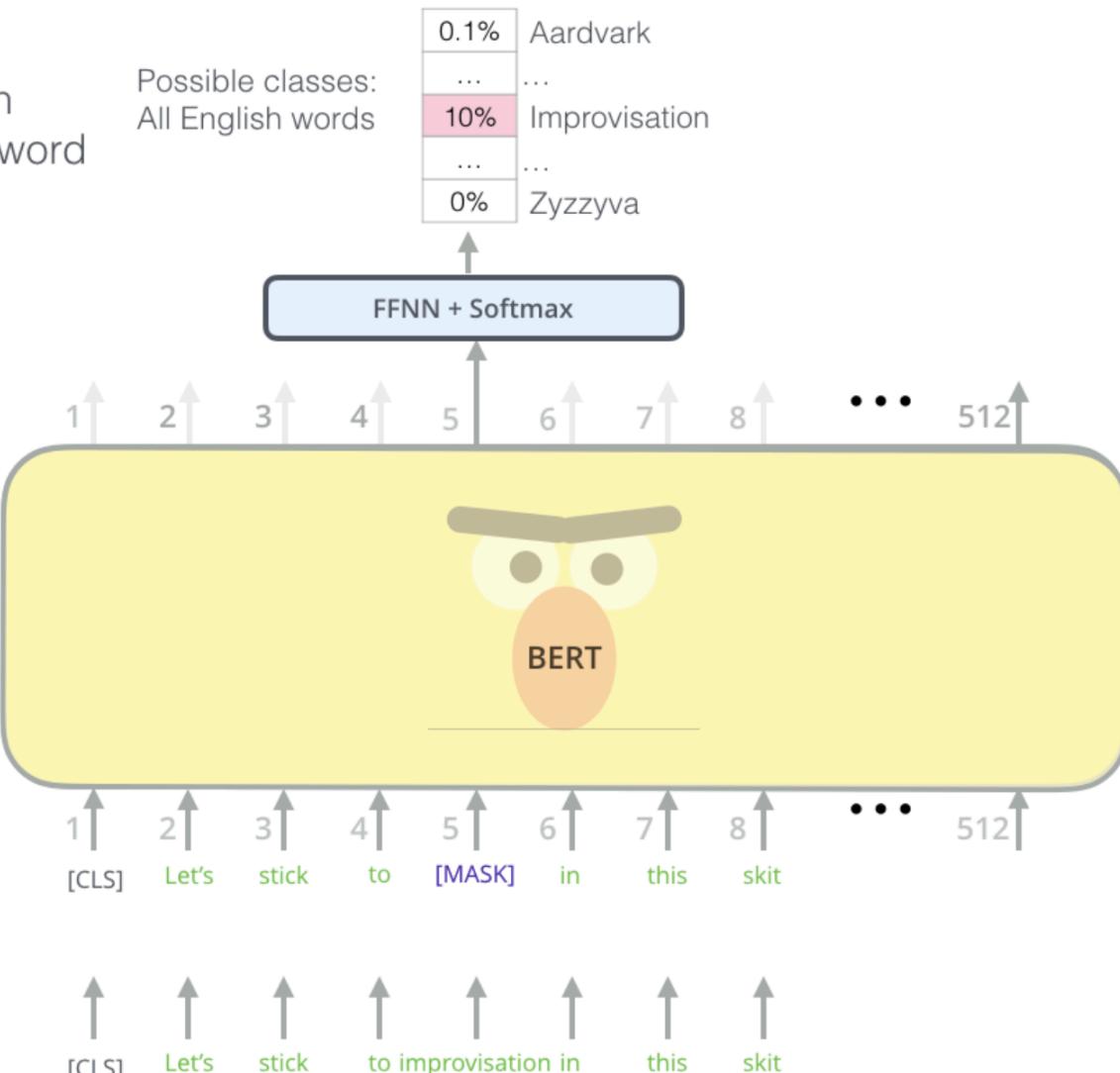
Use the output of the masked word's position to predict the masked word

Possible classes:
All English words

0.1%	Aardvark
...	...
10%	Improvisation
...	...
0%	Zyyzyva

FFNN + Softmax

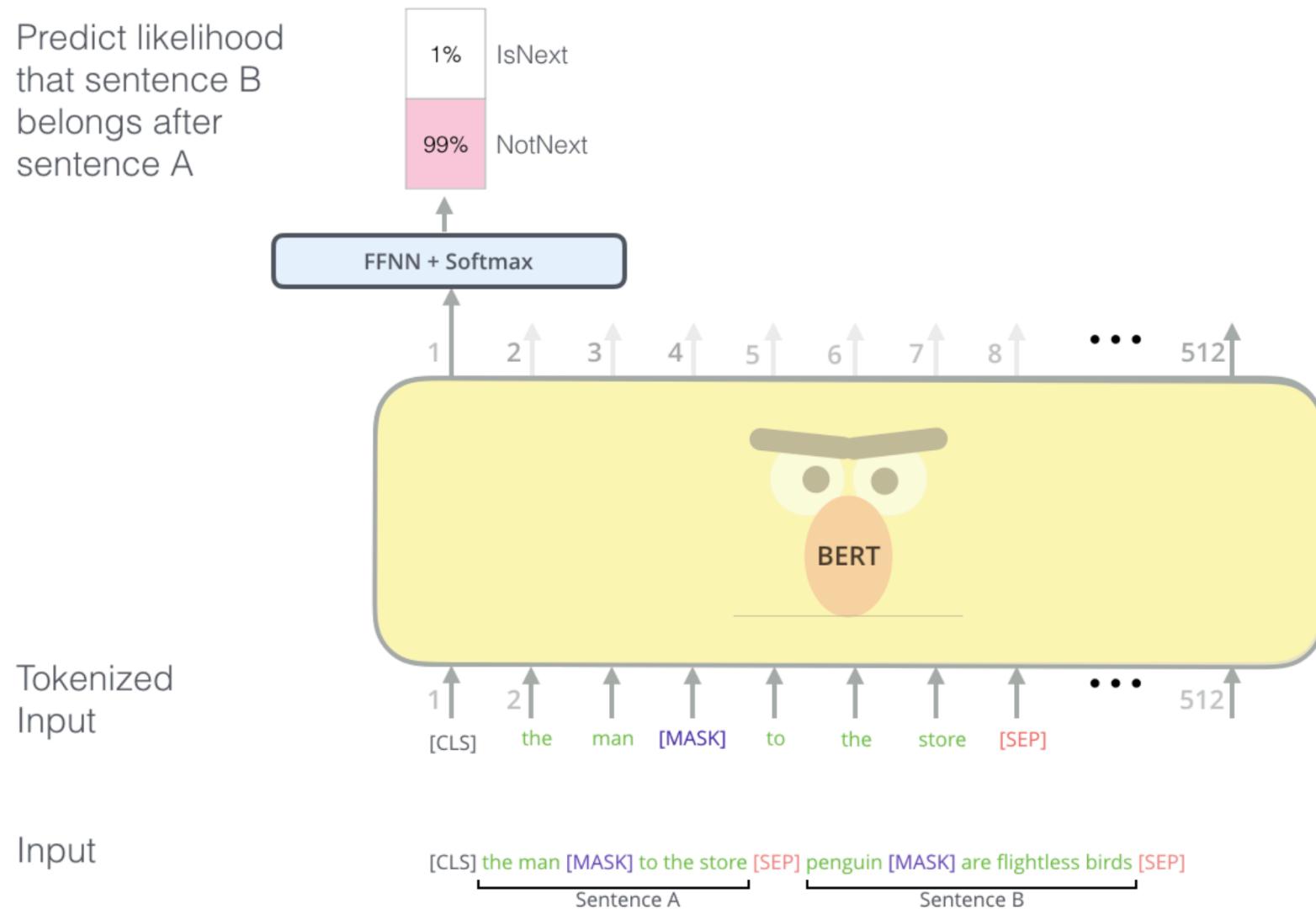
Randomly mask 15% of tokens



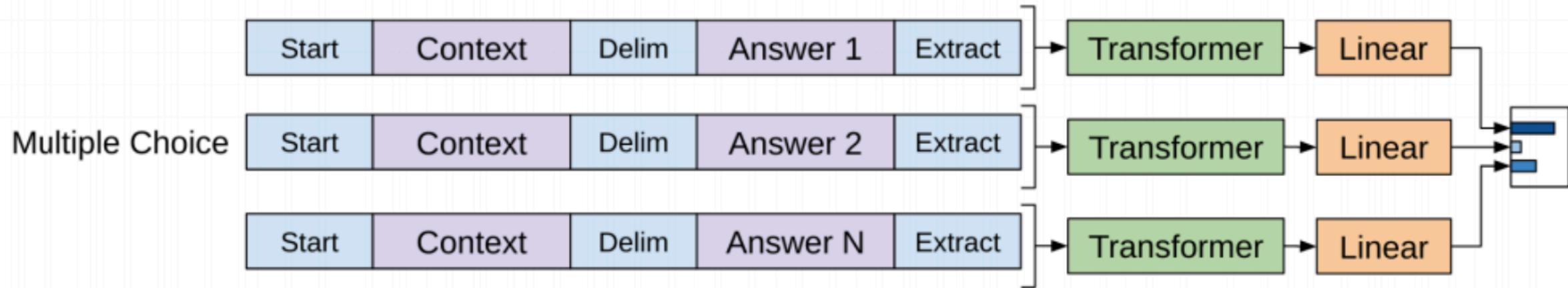
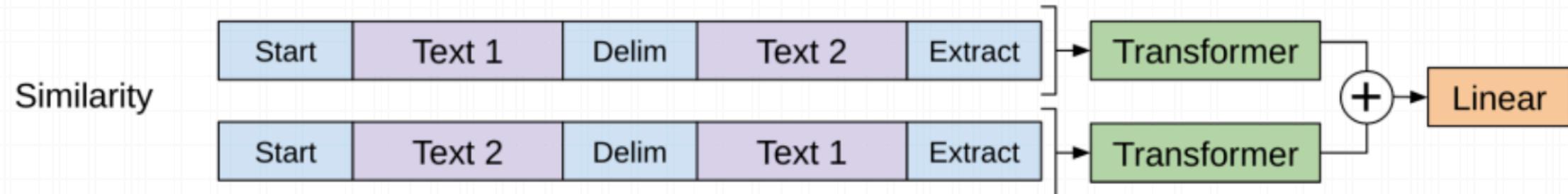
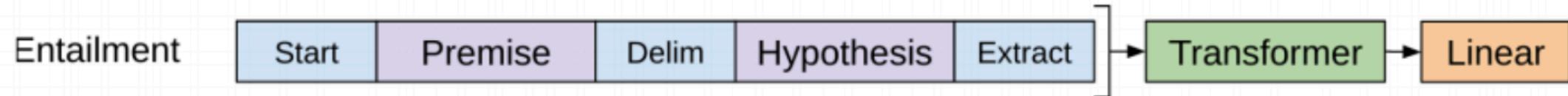
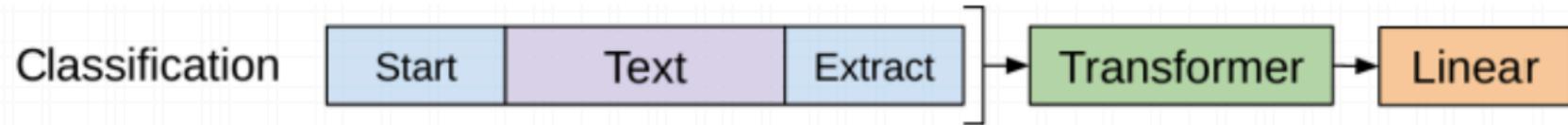
BERT's clever language modeling task masks 15% of words in the input and asks the model to predict the missing word.

BERT- predict is next sentence?

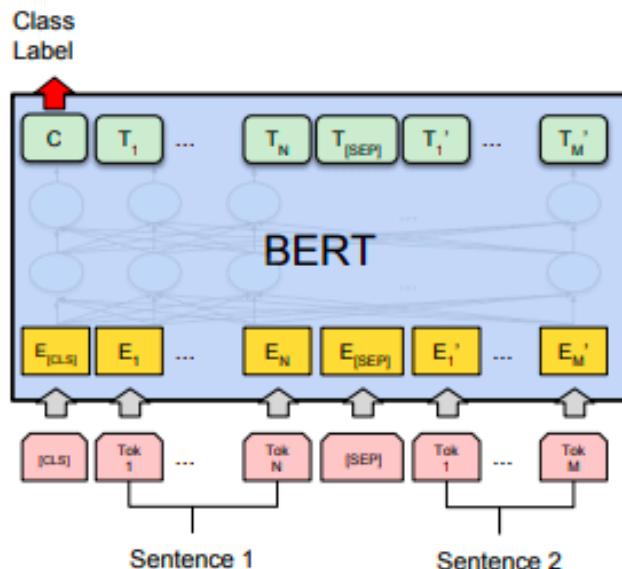
Predict likelihood
that sentence B
belongs after
sentence A



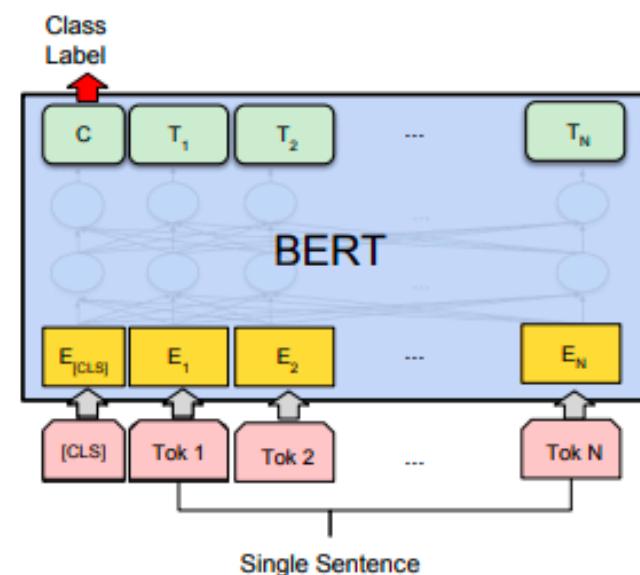
The second task BERT is pre-trained on is a two-sentence classification task. The tokenization is oversimplified in this graphic as BERT actually uses WordPieces as tokens rather than words --- so some words are broken down into smaller chunks.



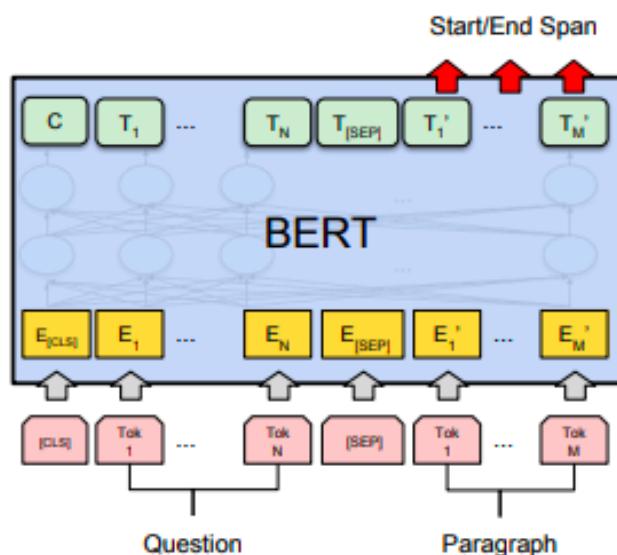
BERT on tasks



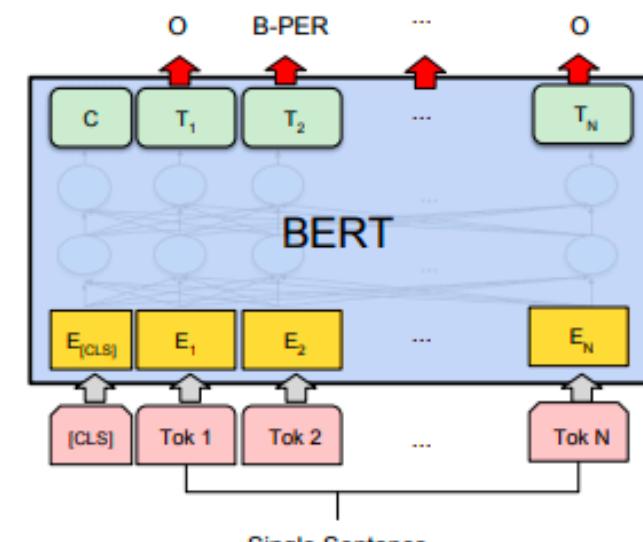
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Take-Aways

- Distributional semantics – learn a word’s “meaning” by its context
- Simplest representation is frequency in documents
- Word embeddings (Word2Vec, GloVe, FastText) predict rather than use co-occurrence
- Similarity measured often using cosine
- Intrinsic evaluation uses correlation with human judgements of word similarity
- Contextualized embeddings are the new stars of NLP